Practical 4: Classification analysis

Garyfallos Konstantinoudis

Spring Term 2026

Part 1: Predicting treatment outcome for breast cancer based on gene-expression

We look back at the dataset which links gene-expression with the response of breast cancer patients to treatment. Our aim is to build a classifier that can discriminate between 2 responder groups (pathologic complete response or minimal residual cancer burden [RCB-I] defining excellent response (coded as 0), vs moderate or extensive residual cancer burden [RCB-II/III] defining lesser response (coded as 1).

For the original publication please see https://jamanetwork.com/journals/jama/fullarticle/899864

Load the dataset including $n = 414$ patients and the predictor matrix including expression of $p = 22,283$ genes.

```
load("../data/JAMA2011_breast_cancer")
#alternatively try load("../../data/JAMA2011_breast_cancer.dms")
y = as.factor(data_bc$rcb)
table(y)
```

```
## y
##   0   1
## 117 297
```

```
x = as.matrix(data_bc$x)
dim(x)
```

```
## [1]   414 22283
```

Question 1.1

Use the sda package to perform a diagonal discriminant analysis (dda). First we rank the features using the sda.ranking function. The sda package jointly estimates the local fdr. How many genes pass a threshold of local fdr < 0.2?

```
library(sda)
```

Question 1.2

Next we use the number of features with local fdr < 0.2 to build a prediction rule for dda and evaluate the prediction rule on the same data (using the same 414 samples as for training the algorithm). How does the confusion matrix (as implemented in the crossval package) look like? What is the sensitivity and specificity of dda?

```
library(crossval)
```

Question 1.3

Compute the area under the curve (AUC) of the receiver operating characteristic (ROC) curve and plot a ROC curve. The package pROC offers the function roc(observed,predicted) which plots the ROC curve and computes the AUC and ROC parameters. Alternatively use the function roc.curve in the PRROC package.

```
library(pROC)
```

Question 1.4

Next we perform linear discriminant analysis (lda) where we first rank the features and define a threshold using local fdr < 0.2 as cut-off point. How many genes are included into the model at a threshold of local fdr < 0.2? Then we build our prediction rule and evaluate it on the same data by computing the confusion matrix, sensitivity and specificity, the AUC and plotting the ROC curve. Which approach performs better, lda or dda?

Question 1.5

Use support vector machines (function svm in the e1071 package) to build a prediction rule and evaluate it on the same data by computing the confusion matrix, sensitivity and specificity, the AUC and the ROC curve.

```
library(e1071)
```

Question 1.5

Use random forests (function randomForest in the randomForest package) to build a prediction rule and evaluate it on the same data by computing the confusion matrix, sensitivity and specificity, the AUC and the ROC curve.

```
library(randomForest)
```

Question 1.6

Which method has the best prediction performance on the dataset? Is it good practice to evaluate the performance of a prediction rule on the same data that was used to build the prediction rule?

Part 2: Evaluating systematically the prediction performance using cross-validation

Next we systematically evaluate the prediction performance using cross-validation as implemented in the crossval package.

```
library(crossval)
```

The first step is to write a function which defines the prediction rule. For example, see below how a prediction rule may be defined for discriminant analysis. This function has the option: diagonal=TRUE which performs dda (assume diagonal covariance matrix and no correlation between predictors) and diagonal=FALSE which fits lda (allow for covariance among predictors). The first step is to rank features and define a cut-off using the local false discovery rate. After feature selection the prediction rule is trained and finally, the prediction performance is evaluated using the confusionMatrix function.

```
predfun.da = function(Xtrain, Ytrain, Xtest, Ytest, diagonal=FALSE)
{
  # estimate ranking and determine the best numVars variables
  ranking.DA = sda.ranking(Xtrain, Ytrain, verbose=FALSE, diagonal=diagonal, fdr=TRUE)
  numVars = sum(ranking.DA[,"lfdr"]<0.2)
  selVars = ranking.DA[,"idx"][1:numVars]

  # fit and predict
  sda.out = sda(Xtrain[, selVars, drop=FALSE], Ytrain, diagonal=diagonal, verbose=FALSE)
  sda.class = predict(sda.out, Xtest[, selVars, drop=FALSE], verbose=FALSE)$class

  # count false and true positives/negatives
  negative = levels(Ytrain)[1] # negatives or baseline is the good response class
  cm = confusionMatrix(Ytest, sda.class, negative=negative)

  return(cm)
}
```

For reproducibility of the results please use the following random number seed.

```
set.seed(2)
```

Question 2.1

Write a similar prediction function for support vector machines.

Question 2.2

Write a similar prediction function for random forests.

Question 2.3

Now use 5-fold cross-validation (K=5) with 2 repetitions (B=2) to evaluate the prediction performance of dda, lda, svm and randomForest. Note that 2 repetitions is just for the interest of time. Ideally you would like to repeat the cross-validation at least 10 times. Which of the four methods has the best classification performance in terms of true positive rate (specificity) and true negative rate (specificity)?

Part 3 (Optional): Predicting credit card fraud

The datasets contains transactions made by credit cards in September 2013 by European cardholders. It is downloaded from https://www.kaggle.com/mlg-ulb/creditcardfraud

The outcome variable is Class, where 1 is a fraudulent transaction and 0 is a valid transaction.

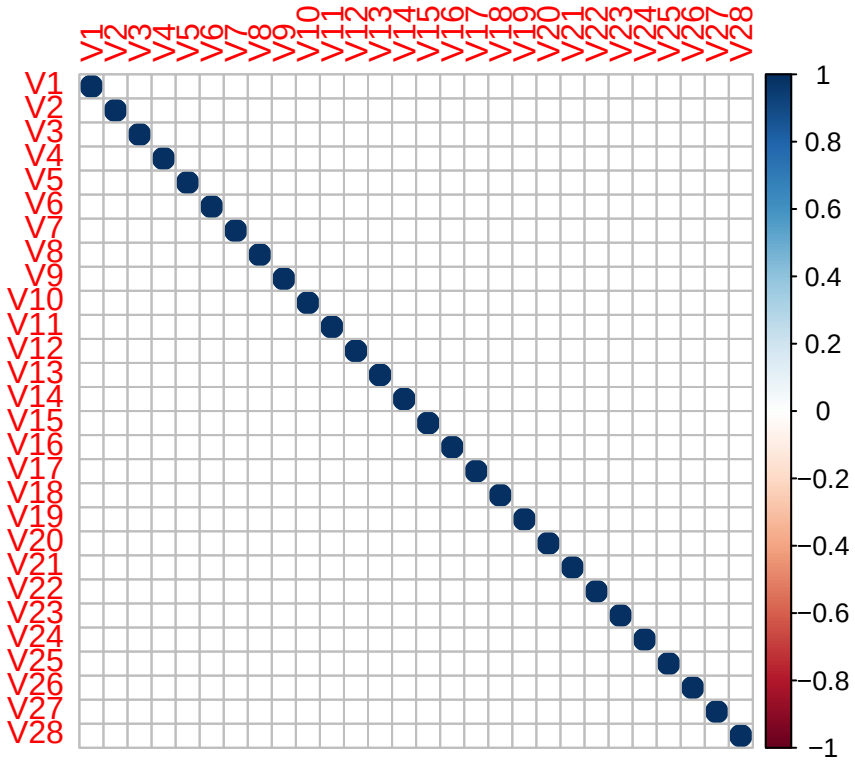
```
cc.data = read.csv("../data/creditcard.csv")
y = as.factor(cc.data$Class)
table(y)
```

```
## y
##      0      1
## 284315    492
```

Please note that fraudulent transactions are very rare events.

The aim of this exercise is to predict credit card fraud from a set of 28 input features. These are principle components of the original features. The original features and more background information cannot be provided due to confidentiality issues. One advantage of working on principle components is that the input features are perfectly uncorrelated.

```
x = as.matrix(cc.data[,2:29])
library(corrplot)
corrplot(cor(x))
```



For reproducibility of the results please use the following random number seed.

```
set.seed(3)
```

This is a pretty tall dataset with many observations. If you are impatient for support vector machines and random forest to run you may think of down-sampling the dataset to test your algorithm by drawing 10,000 random samples.

```
#randomNumber = sample(1:length(y), size=10000, replace = FALSE)
#y=y[randomNumber ]
#table(y)
#x=x[randomNumber,]
```

Question 3.1

Construct a classification rule based on dda. Since there is no correlation between features we do not need to explore lda as well. For 28 features it is not possible to fit the local fdr because these are too few variables. Given the large sample size we may include all features into the model.

Question 3.2

Which summary-statistic would you use to evaluate the performance of the classification in this particular application?

Question 3.3

Write the equivalent prediction function for dda without variable selection (modify predfun.da() from question 2) and evaluate it using cross-validation (K=5, B=1). Use an appropriate summary statistic to measure the performance of the classification of rare events.

Question 3.4

Perform cross-validation using the prediction functions developed for svm and randomForest in Question 2.1 and Question 2.2. Which method has the best prediction performance?